Patent
110630-017

UNITED STATES PATENT APPLICATION
FOR

SYSTEMS AND METHODS FOR IMAGE PATTERN RECOGNITION

INVENTORS:

OLE EICHHORN
DIRK G. SOENKSEN

PREPARED BY:
PROCOPIO, CORY, HARGREAVES & SAVITCH LLP
530 B STREET, SUITE 2100
SAN DIEGO, CALIFORNIA 92101-4469

Express Mail Number:
EL986040437US

SYSTEMS AND METHODS FOR IMAGE PATTERN RECOGNITION

Related Application

[01] This application claims priority to United States Provisional Patent Application, serial number 60/451,083 filed on February 28, 2003, which is incorporated herein by reference in its entirety.

Background

1. Field of the Invention

[02] The present invention generally relates to automated inspection of digitized microscope slides and more particularly relates to pattern recognition techniques employed in automated inspection of virtual slides.

2. Related Art

[03] An obstacle to automating microscopic inspection has been the inability to efficiently digitize entire microscope specimens at diagnostic resolutions. Conventional approaches for creating virtual slides have relied on image tiling. Image tiling involves the capture of multiple, small regions of a microscope slide using a traditional charge coupled device ("CCD") camera. These tiles are typically "stitched" together (aligned) to create a large contiguous digital image (mosaic) of an entire slide. A minimum of 2,250 individual camera tiles are required to digitize a typical 15 mm x 15 mm area of a slide at 50,000 pixels/inch (0.5 μm/pixel).

[04] Image tiling carries several disadvantages. First, images frequently have distortion because image tiles are limited to a single focal plane from the camera's fixed area objective lens. Second, an image tiling system produces optical aberrations that are circularly symmetric about the center of the image tile. Third, full pixel resolution is

usually unavailable because color CCD cameras lose spatial resolution upon interpolation of color values from non-adjacent pixels.

[05] Conventional approaches to pattern recognition are also cumbersome. Despite many years of improvements in optical microscopy, in most cases a human operator is still required to manually evaluate a specimen through the eyepieces of a dedicated instrument. Worldwide, in thousands of clinical laboratories, more time is spent performing manual microscopy than any other in-vitro diagnostics testing procedure. In an estimated 20,000 research laboratories, manual microscopic inspection is an essential tool for screening drug targets and for conducting toxicology studies. While offering immense opportunities for automation, microscopic inspection remains a bastion of manual labor in an environment that is otherwise converting to automated solutions.

[06] Some attempts at computer aided pattern recognition have been made. These approaches to pattern recognition in microscopic images are based on morphological features. A large number of feature metrics (e.g., cell size, nuclear-to-cytoplasmic ratio, roundness, density, color, texture, etc.) are computed to identify "objects" (e.g., cells) that appear in the image. Pattern recognition is achieved by correlating the feature metrics for an unknown object with those of a known object. Feature based approaches have had some success in cytology, where a high level of cell diversity is required to allow objects to be identified. These conventional approaches, however, are immensely complicated or completely untenable for histology imagery data where reliable object segmentation is difficult.

Summary

[07] Systems and methods for image pattern recognition comprise digital image capture and encoding using vector quantization ("VQ") of the image. A vocabulary of vectors is built by segmenting images into kernels and creating vectors corresponding to each kernel. Images are encoded by creating vector index files with indices ("pointers") to the vectors stored in the vocabulary. The vector index files can be used to reconstruct the images by looking up vectors stored in the vocabulary. Pattern recognition of candidate regions of images can be accomplished by correlating image vectors to a pre-

trained vocabulary of vector sets comprising vectors that correlate with particular image characteristics. In virtual microscopy, the systems and methods are suitable for rare-event finding, such as detection of micrometastasis clusters, tissue identification, such as locating regions of analysis for immunohistochemical assays, and rapid screening of tissue samples, such as histology sections arranged as tissue microarrays (TMAs).

Brief Description of the Drawings

[08] The details of the present invention, both as to its structure and operation, may be gleaned in part by study of the accompanying drawings, in which like reference numerals refer to like parts, and in which:

[09] **Figure 1** is a block diagram illustrating an example process for encoding and decoding an image using vector quantization according to an embodiment of the present invention;

[10] **Figure 2** is a flow diagram illustrating an example method for image pattern recognition according to an embodiment of the present invention;

[11] **Figure 3** is a block diagram illustrating an example client/server implementation according to an embodiment of the present invention;

[12] **Figure 4** is a block diagram illustrating an example nested kernel logic according to an embodiment of the present invention;

[13] **Figure 5** is a flow diagram illustrating an example process for creating variable sized kernels according to an embodiment of the present invention;

[14] **Figure 6** is a table diagram summarizing a representative sample of slides used for a study according to an embodiment of the present invention;

[15] **Figure 7** is a table diagram summarizing the results of a plurality of slides from a study according to an embodiment of the present invention;

[16] **Figure 8** is a graph diagram illustrating an example change in vocabulary size as slides were processed during a study according to an embodiment of the present invention;

[17] **Figure 9** is a graph diagram illustrating an example change in index size as slides were processed during a study according to an embodiment of the present invention; and

[18] **Figure 10** is a block diagram illustrating an exemplary computer system as may be used in connection with various embodiments described herein.

Detailed Description

[19] Certain embodiments as disclosed herein provide for pattern recognition systems and methods. For example, one method as disclosed herein allows for a digitized image to be segmented into a plurality of image kernels. The image kernels are then analyzed to determine the intensity of each pixel in the kernel. The pixel intensities for an image kernel are then combined into a vector and the vector is stored in a data structure with other vectors for the image. Each vector is indexed to create a composite index for the digitized image that can later be reconstructed using the indices and the vectors.

[20] After reading this description it will become apparent to one skilled in the art how to implement the invention in various alternative embodiments and alternative applications. However, although various embodiments of the present invention will be described herein, it is understood that these embodiments are presented by way of example only, and not limitation. As such, this detailed description of various alternative embodiments should not be construed to limit the scope or breadth of the present invention as set forth in the appended claims.

[21] A first step in automating microscopic inspection is to create a virtual slide, i.e., a high resolution digital image of an entire slide. The systems and methods of the present invention overcome many of the disadvantages of the existing image pattern recognition techniques by employing a technique known as vector quantization ("VQ"). VQ is a computational technique for encoding bitstreams using a vocabulary. VQ is particularly suitable for encoding images that have significant redundancy.

[22] VQ is a useful approach for pattern recognition in imagery data that is repetitive in nature. Fundamentally, pattern recognition using VQ is based on comparing previously encoded patterns (i.e., particular characteristics) with new imagery data. The computational challenges of organizing and retrieving significant volumes of imagery data are immense and prior to the present invention have prevented the application of VQ to pattern recognition in large images. The implementation of VQ in accordance with the

systems and methods described herein overcomes these computational challenges and makes pattern recognition in virtual microscopic images feasible.

[23] The systems and methods for VQ-based image pattern recognition of the present invention represent a significant improvement over the prior art for several reasons: First, VQ-based pattern recognition is efficient because encoding of image information is performed once, after which pattern matching can be performed solely by correlating VQ vector information, rather than comparison of image information. Second, VQ encodes all image characteristics without need for predetermined heuristics that identify significant features. Pattern matching can be morphometric, colorimetric, context-dependent, etc., depending upon the training used to create vectors of interest. Finally, VQ pattern recognition learns automatically as it is used. Statistical methods determine the significance of vectors that are characteristic for particular patterns.

[24] Figure 1 illustrates a process 100 for encoding an image using VQ. Process 100 begins after digital image 105 is analyzed and segmented into small rectangular regions called image kernels 108. Image 105 can be, for example, a liver sample comprising one or more image characteristics. An image characteristic is any user-defined attribute that may be exhibited by a region of an image, such as a cluster of tumor cells or region of analysis ("ROA") within a tissue microarray ("TMA") spot. Kernels can vary in size from 1 to 64 pixels per side, or 1 to 4096 total pixels per kernel. Larger kernel sizes may also be employed as will be understood by those skilled in the art.

[25] Next, a vector 112 corresponding to each kernel 108 is generated and stored in vocabulary 110. Vocabulary 110 is a data structure comprising the vectors 112 corresponding to kernels 108. The data structure may be a database, a file, a linked list, or some other collection of vectors 112. A vector 112 is a one-dimensional array of scalars. In vector quantization, each vector represents the pixel intensities of all pixels in a kernel, for each of three color channels. For example, a kernel comprising 15 pixels by 15 pixels corresponds to a vector with 225 * 3 = 675 intensity values.

[26] Vector index file 120 is a collection of indices 116 that correspond (i.e., "point") to the vectors 112 that comprise image 105. Each vector 112 in vocabulary 110 is

assigned a unique index 116. In one embodiment, an index 116 is a unique 4-byte integer. Once an image is completely segmented into a plurality of kernels 108, corresponding vectors 112 are generated for each kernel 108 and the vectors 112 are indexed. At this point, the image is said to be vector quantized, or VQ-encoded. After image 105 has been VQ-encoded, it is completely represented by a series of indices 116 in vector index file 120 (together with the contents of the vocabulary 110).

[27] Image 105 can be reconstructed (i.e., VQ-decoded) using the indices 116 to lookup each vector 112 in vocabulary 110. Decoding uses the vectors 112 to recreate the corresponding kernels 108 necessary to render image 105.

[28] Capitalizing on redundant kernels is a significant component of VQ-encoding. During the VQ-encoding process, each newly identified kernel 108 is compared to the vectors 112 already stored in the vocabulary. If a sufficiently close match is found with a vector 112 already in vocabulary 110, then the vector 112 for that newly identified kernel is re-used. If, however, a sufficiently close match is not found with a vector 112 already in vocabulary 110, then the vector 112 for that newly identified kernel is appended to vocabulary 110. As a result, over time vocabulary 110 is populated with nearly all kernels 108 which are found in the imagery data for a particular image type, e.g., samples of liver tissue. Thus, as slides of a given tissue type are encoded, more and more of the kernels 108 which appear in such slides are stored in the vocabulary 110, and correspondingly fewer new vectors 112 need be added. In one embodiment, a mature vocabulary 110 may VQ-encode an image 105 without adding any new vectors 112 to the vocabulary 110.

[29] Searching vocabulary 110 for pre-existing vectors to reuse is computationally expensive. The systems and methods of the present invention achieve, by several orders of magnitude, a reduction in processing time and memory usage. One technique for accomplishing this improved processing efficiency includes the computation of a correlation between a vector and its corresponding image characteristic, as described in the next figure.

[30] Figure 2 is a flow diagram illustrating a method for image pattern recognition according to an embodiment of the invention. The image pattern recognition process can be divided into a training stage 201 and a pattern recognition stage 202. In the first step 205 of the training stage 201, a human expert identifies a region of an image as characteristic of an attribute. For example, an expert in the biological sciences might examine a tissue sample and visibly identify a group of tumor cells or ROAs within a TMA. Typically the attribute is recognized by the expert without regard to the specific visual characteristics which comprise the attribute, e.g. morphometric, colorimetric, and/or context-dependent features of the image.

[31] In step 210, a vector set is created to store each vector index that corresponds to a vector found in the image characteristic. Multiple vector sets can be created for different characteristics. For example, different vector sets can be created for different tumor types identified by experts in step 205. For instance, one vector set might be developed for melanoma and another for leukemia.

[32] In step 215, the correlation of each vector to the image characteristic is computed. In one embodiment, the correlation is computed using the following ratio:

$$\frac{\text{Number of times the vector is observed in the region exhibiting the characteristic}}{\text{Number of times the vector is observed overall}}$$

[33] The correlation of each vector to the image characteristic is stored in the vector set along with the corresponding vector index in step 220. Vector sets contain vector indices together with their associated correlation to the previously identified image characteristic. In step 225, the training stage is repeated for the next expertly identified image. As a result of a series of images passing through training stage 201, a correlation between individual vectors and a particular vector set will be computed. The more images used to train the system, the more statistically sound the correlation result will be for an individual vector and a particular vector set. When a sufficiently large number of images have been processed and the individual vectors correlated with a particular

characteristic represented by a vector set, the training stage will be complete and pattern recognition can begin.

[34] If a particular vector is only ever found in regions of images exhibiting the characteristic, then the correlation between the vector and the characteristic will have a maximum value of one (1), which is 100% correlation. In this case, the vector will have a high predictive value for the characteristic. On the other hand, if a particular vector is never found in regions of images exhibiting the characteristic, the correlation will have a minimum value of zero (0), which is 100% non-correlation. In this case, the vector will have a high predictive value for the absence of the characteristic. Finally, if a particular vector is sometimes found in regions exhibiting the characteristic and sometimes it is not, then the correlation value (also referred to as "ratio") will be somewhere between 0 and 1. If the ratio is near one-half, the vector will have a low predictive value for the presence of absence of the characteristic.

[35] In the first step of the pattern recognition stage, step 230, a candidate region of a new image is identified. A candidate region is a region of an image that has not been expertly identified to possess any regions containing any image characteristic of one or more attributes. In many cases the candidate region of an image is the entire image, or that portion of the image which contains sample tissue. As such, the vectors corresponding to the candidate region may or may not have been encountered during the training stage. In step 235, the average of the correlations to a particular vector set corresponding to the vectors in the candidate region is computed. In step 240, the average is evaluated to determine the likelihood that the candidate region exhibits the characteristic represented by one or more vector sets. High average correlations indicate a high probability that the candidate region exhibits the characteristic, while low average correlations indicate a low probability. The candidate region may be compared to multiple vector sets to determine the probability that the region exhibits various different characteristics. For example, an image of histology tissue may be compared to various vector sets which represent the visual characteristics of different types of tumors, to determine the likelihood that the tissue contains a tumor, and if so, the type of tumor. A

significant advantage of the described technique for pattern recognition is that not only are candidate regions classified by characteristics they are likely to exhibit, but the actual probability that they do so is computed.

[36] In one embodiment, the systems and methods for image pattern recognition are implemented using a distributed client/server architecture. Figure 3 illustrates a client/server embodiment of the present invention. Server 302 comprises the VQ-encoding server software 308 and vocabulary 310, and server 302 communicates with one or more clients 301 over a network 303, for example a LAN/WAN communications channel. Clients 301 comprise the VQ-encoding client software 307, which process image files 305 to generate vector index files 320. In this embodiment, vocabulary 310 is located centrally on server 302, providing shared access to clients 301. Vocabulary centralization facilitates database maintenance and eliminates the need to synchronize distributed vocabularies as additional vectors are appended. In addition, a client/server implementation permits the concentration of computationally intensive tasks, such as searching and indexing, to be performed at a central location, relieving clients 301 from the burden.

[37] For large vocabularies the time required to index vocabulary 310 takes about twice as long as the time required for VQ-encoding a typical image 305. For example, out of a total time of 60 minutes to process an image, the indexing time would be 60 minutes while the VQ-encoding time would be 20 minutes. Of the 20 minutes required for encoding, segmentation of image 305 into kernels requires about 5 minutes, and searching vocabulary 310 requires about 15 minutes. When implemented using the client/server embodiment, a server process 308 can be made continuously available with a pre-indexed vocabulary. The encoding process for any given image will not require re-indexing the vocabulary, eliminating that portion of the processing time. The client/server communication overhead is relatively small compared to the overhead of subdividing the image 105 into kernels (on the client side) and searching the vocabulary 310 for these kernels (on the server side). Thus, dividing the processing between two distributed computers does not add appreciably to the execution time. On the other hand,

by distributing the processing, the client-side kernel subdivision may be overlapped against the server-side vocabulary searching, resulting in a net reduction of overall processing time by eliminating the client processing time from the critical path. In this way, an overall encoding time of 60 minutes per slide can be reduced to 15 minutes per slide.

[38] Division of processing labor between client and server in this embodiment has other efficiencies. Indexing and searching the vocabulary 310 is a computer-intensive task, best performed by fast processors equipped with large amounts of memory. Segmenting images 305 into kernels and communicating over the network 303 requires comparatively less processing power. This embodiment enables a small number of powerful server machines 302 to perform processing on behalf of a large number of less-powerful client machines 301, yielding an overall increase in the efficiency of the system. Additionally, vocabulary files 310 are typically quite large, and are often stored on large disk arrays (e.g. RAID), while vector index files 320 are several orders of magnitude smaller, and can be stored on ordinary computer disks.

[39] In addition to a client/server embodiment, various other aspects of the systems and methods for image pattern recognition can be employed to improve both the efficiency and accuracy. In one embodiment, for example, encoder 307 can be programmed to transform the entire image from the RGB ("red-green-blue") color-space to the YCRCB ("intensity-red-blue") color space prior to encoding. As kernels are matched to vectors in the vocabulary 310, the Y channel values are matched more carefully than the C channel values, because human perception of image resolution relies primarily on differences in intensity, rather than differences in color. This prioritization enables lower tolerances in certain comparisons without compromising the information content of the kernels. By allowing the lower tolerances, more correllations are found. Accordingly, the vocabulary 310 is smaller, which results in a decrease in size for the vector index 320, and index search times are therefore reduced.

[40] In another aspect of the invention, color channel optimization can be implemented to further reduce the size of vocabulary file 310. Human perception of differences in

image intensity values is non-linear; it is most sensitive in the middle of the range (moderate intensity), and least sensitive at the extremes (bright or dark). Encoder 307 can be configured to take advantage of this so that as kernels are matched to vectors in the vocabulary 310, Y channel values in the middle of the intensity range are matched with a closer tolerance than Y channel values at the ends of the range. This prioritization similarly enables lower tolerances in certain comparisons without compromising the information content of the kernels 108. As discussed above, the lower tolerances for certain less critical comparisons advantageously results in a significantly smaller size vocabulary 310, thereby significantly reducing index search times.

[41] In yet another aspect, differential encoding can be employed for reducing the bit-size of C channel values within the YCRCB color space. With differential encoding, each intensity value within a vector is replaced by the difference between the intensity value and the average value of the three nearest neighbors to its upper left. For example, if the value represents the pixel at coordinate position [x, y], the average is computed of the values for pixels [x-1, y], [x-1, y-1], and [x, y-1]. (Note that in the standard convention for digital images, [0, 0] is the upper left corner of the image, with x increasing to the right, and y increasing downward.) In addition, for the C channel intensity values, intensities are typically stored as 8-bit values, but the difference between adjacent intensities can be represented by 4-bit values without loss of fidelity. This reduces the range of each value from 256 (for intensity) to 16 (for differential). Accordingly, a substantial reduction in the size of the index trees for each value is gained, thereby yielding faster searching of vocabulary 310.

[42] The data structure used to search vocabulary indices by the encoder 308 is an important determinant of system resource usage and encoding performance. In one embodiment, a new data structure called a vector tree can be used to store and search vector indices. A vector tree comprises a balanced tree wherein each leaf node is the root of another tree. The top-level tree represents the first dimension of each kernel. The leaf nodes of this tree contain trees which represent the second dimension of each kernel, for each defined value of the first dimension. The leaf nodes of each second-level tree

contain trees which represent the third dimensions, and so on. In addition to containing the root for the next-level tree, each leaf node may also represent the last dimension of a defined vector. In this case the vector's index is also stored in the leaf node. This arrangement enables multi-dimensional vectors of different sizes to be stored together in one structure, facilitating concurrent searching for multiple kernels having different dimensions.

[43] In one embodiment, nested kernel logic is used to store vector intensity values more efficiently and thereby reducing index search times. Kernels of size k x k pixels have $3k^2$ intensity values. The size of these vectors is a key driver of the size of the vocabulary file 310, as well as the size of vocabulary indices and the time required for vector index searches.

[44] In one embodiment, kernels are represented in nested fashion which reduces the number of intensity values required for each kernel to 6k. Figure 4 is a diagram illustrating nested kernel logic that can be used to store vectors. Using this nested kernel logic, each kernel 408 of size k ("outer kernel") is assumed to have a previously located kernel 405 of size k-1 inside it ("inner kernel"), with their upper-left corners aligned. In actual practice this always occurs, because the encoding logic tries searching the vocabulary for successively larger kernels at a pixel location until a "not found" occurs, as described below with reference to variable sized kernel logic.

[45] This enables the outer kernel 408 to be represented by 6k (i.e. 3(2k)) values. The factor of three is from the three color channels. Of the 2k, the first value is represented by the index of the inner kernel 405 and the next k values are the pixels 410 along the right edge of the outer kernel 408. The remaining k-1 values are the pixels 415 along the bottom edge of the outer kernel 408.

[46] Representing kernels with 6k intensity values instead of $3k^2$ intensity values provides a significant reduction in the number of valued needed to represent a vector. For example, for a kernel of size k=15 pixels, there would be 90 values to store (instead of 675) for that kernel. This dramatically reduces the size of the vocabulary file and the

corresponding vector trees used by the VQ-encoder for index searching. The encoding search times are similarly reduced.

[47] Referring back to Figure 1, another technique for decreasing search time involves VQ-encoding an image 105 using variable-size kernel logic. Exemplary kernels 108 comprising VQ-encoded image 105 in Figure 1 are variable in size. This technique enables mapping of image data by kernels of varying size, accommodating a wide range of objects, structures, and features in the encoded images. It also removes the need for specifying or deriving an optimum kernel size for a given image.

[48] Figure 5 is a flow diagram illustrating an example process for creating variable sized kernels according to an embodiment of the present invention. In step 505, the VQ-encoder, iterates through each pixel location from upper left to lower right. In step 510 (performed for each pixel location), the vocabulary is searched for a matching kernel of size n (a parameterized value for the smallest kernel size, typically 3 or 4). In step 515, if the kernel is found, n is incremented in step 520 and the search for a kernel of size n+1 is invoked to try to find an existing kernel of size n x n that matches the identified square region. The encoder loops through the vocabulary trying sequentially larger kernels until a "not found" condition occurs in step 515.

[49] In step 525, if a match between the identified image location and an existing n x n kernel is not found, then the identified square region is mapped to the existing kernel of size n-1 x n-1 that was found in the previous iteration of the loop. The use count for the n-1 x n-1 kernel will be incremented to indicate that another instance of the kernel is being used to map a region of the image. As for the n x n region that was searched but never found, in step 530 that region will be added to the vocabulary (and appropriately indexed) and the use count for the new kernel will be initialized to one. Each time a kernel is matched, the use count is incremented, and when the use count exceeds a parameterized threshold (typically 20) the kernel may be used for image encoding. In this way the vocabulary is continuously enlarged over time with ever-larger kernels which contain ever-more image information.

[50] The above described embodiment describes the use of variable size square kernels. In another embodiment, rectangular kernels may also the employed by incrementing the size kernel to be searched for to n+1 x n, rather than n+1 x n+1. For example, in step 515, if the kernel is found, the size of one side of the kernel is incremented in step 520 and the search for a kernel of size n+1 x n is invoked to try to find an existing kernel of size n+1 x n that matches the identified rectangular region. The encoder loops through the vocabulary trying sequentially larger rectangular or square kernels until a "not found" condition occurs in step 515.

[51] In an exemplary embodiment, a feasibility study was conducted using liver histology sections to determine the performance and storage requirements of systems and methods described herein. One goal of this study was to verify that VQ-encoding of virtual microscope slide images could be performed on readily available computing hardware in run times that are compatible with practical applications. Another goal was to determine how quickly vocabulary sizes and corresponding index sizes converge. Convergence occurs as slides of a given tissue type are encoded. Over time, as more and more of the kernels which appear in such slides are stored in the vocabulary, correspondingly fewer new vectors need be added.

[52] The study was performing using 49 liver tissue slides. These slides were standard H&E-stained histology samples. The slides were scanned with a ScanScope® T108 scanner. A virtual slide TIFF file was created for each slide at 54,000 pixels per inch (0.5μm per pixel) using a Nikon 20x/0.75 Plan Apochromat objective lens, and compressed at a compression ratio of 20:1 using Aperio Technologies, Inc.'s JPEG2000 compression software using 7/9 wavelets. In compressed form, each virtual slide TIFF file comprised approximately 250MB. Figure 6 is a table that summarizes a representative sample of the slides used for this study.

[53] The method used in the study employed a tolerance parameter to determine how closely kernels from an image must match vectors in the vocabulary. The study was done three times with three different values for the tolerance parameter: 4% (high), 5% (medium), and 6% (low). In each case, the vocabulary was initially empty. In other

words, the three studies began with no pre-existing kernels in the vocabulary. Each of the slide images was encoded against the vocabulary yielding a vector index file. The encoding process progressively added kernels to the vocabulary. Each slide was decoded after encoding and visually inspected using Aperio Technologies' ImageScope™ viewer to verify that the encoding/decoding process was successful.

[54] The slides were encoded using a Dell Dimension 700MHz Pentium III desktop computer with 1GB RAM and 80GB hard disk. This computer ran Windows 2000 Professional with all unneeded operating system services disabled. Timing, disk and memory measurements were performed by the VQ-encoding software itself.

[55] A sample of the results from the first study is shown in the table in Figure 7. This table shows processing statistics for the first five slides processed, and for the last five slides processed (out of 49 total slides). The last three columns show the processing times. The average index load time was 30 minutes, which increased continuously throughout the study to a maximum of 39 minutes. The average encoding time was 17 minutes with a max of 24 minutes. The overall processing time averaged 48 minutes, with a maximum total time of 63 minutes.

[56] A key observation from this study was whether the vocabulary size would converge, that is, reach a point where successive new slides added fewer and fewer kernels to the database. The cumulative total of vectors column (#vectors) in Figure 7 shows that the first few slides added significantly more vectors to the vocabulary than the last few slides. After 49 slides, a substantial amount of convergence had taken place. Figure 8 is a graph which shows the change in vocabulary size as slides were processed. Although the vocabulary never reached an asymptote, the rate of growth after 49 slides substantially slowed.

[57] Another key observation from this study was whether the index size would "converge", and how large the memory-resident index would be. The size of vector index column (idx MB) in Figure 7 shows that the initial index size was 520MB after only one slide, but thereafter the growth slowed considerably and nearly stopped altogether. Figure 9 is a graph which shows the change in index size as slides were

processed. The index size never exceeded the physical memory available on the machine (1GB), so paging was never required. If the index cannot remain memory-resident encoding performance would decrease dramatically. Figure 9 also shows the index processing time increasing as slides are processed. In the study, the index load time did not stop growing as fast as the index size because as the total index becomes larger each new index requires more time to insert.

[58] Micrometastasis is a practically useful rare-event finding application that can benefit greatly from the systems and methods of the present invention. For patients with tumors, a key question is whether cancer cells are metastasizing, i.e. breaking free from the tumor and traveling to other areas of the body through the bloodstream. Micrometastasis is the presence of small clusters of 1-10 cancer cells in blood. To determine whether such metastasis is taking place, up to fifty cytology slides are prepared with a patient's blood. These slides are then examined with a microscope by a pathologist to determine if micrometastasis clusters are present.

[59] An enrichment procedure is typically used to increase the occurrence of potential tumor cells in a block sample. Tumor cells are differentiated from normal cells based on the following morphological features:

a) Size: typically at least 20μm diameter (normal white blood cells are 10μm-15μm).
b) Nuclear size: Large and usually round (normal nuclei are smaller and/or irregular in shape).
c) Nuclear morphology: Usually one or more nucleoli are present (small dark blue staining regions within the nuclei).
d) Nuclear staining: Usually stained lighter (less hematoxylin) compared to a normal white blood cell.
e) Cytoplasm staining: May be stained bluer (more basophilic) than a normal white blood cell.
f) Nuclear to cytoplasmic ratio: Usually higher than normal white blood cell indicating a larger nucleus.

[60] Differentiation based on these attributes is subtle and requires an experienced pathologist for detection. In this type of rare event detection application it is important not to have false negatives, i.e., it is important not to miss micrometastasis clusters despite their sparse occurrence and difficulty of detection.

[61] To reduce the probability of false negatives, slides are treated with an immunocytochemical antibody that is specific to epithelial cells (a type of cell from which tumors frequently derive). The antibody is prepared as a stain which colors cells of epithelial origin red, whereas normal white blood cells remain unstained. This significantly increases the contrast of micrometastasis clusters, simplifying visual detection and reducing the probability of false negatives. With this stain, micrometastasis slides samples can be scanned at a lower resolution, enabling scanning to be performed more quickly.

[62] To further reduce the probability of false negatives, up to fifty slides are visually scanned, and in a typical case 1-10 clusters occur per slide, a very sparse distribution, and it is important for each cluster to be found to ensure a reliable diagnosis. Some false positives may also occur, for example, dust particles which absorb the immunocytochemical stain and appear as red globs, and other debris in the blood sample. All located potential clusters must be examined visually at high magnification to distinguish false positives from genuine micrometastasis clusters.

[63] A threshold is required to determine when a region constitutes a found target. The method can be run iteratively with different threshold values to determine the value which yields the best results. In practical application this threshold would be kept on the low side of optimal to bias the pattern recognition in favor of false positives (reducing the possibility of false negatives).

[64] Fig. 10 is a block diagram illustrating an exemplary computer system 550 that may be used in connection with the various embodiments described herein. For example, the computer system 550 may be used in conjunction with a server computer or client computer previously described with respect to Fig. 3. However, other computer systems and/or architectures may be used, as will be clear to those skilled in the art.

[65] The computer system 550 preferably includes one or more processors, such as processor 552. Additional processors may be provided, such as an auxiliary processor to manage input/output, an auxiliary processor to perform floating point mathematical operations, a special-purpose microprocessor having an architecture suitable for fast execution of signal processing algorithms (e.g., digital signal processor), a slave processor subordinate to the main processing system (e.g., back-end processor), an additional microprocessor or controller for dual or multiple processor systems, or a coprocessor. Such auxiliary processors may be discrete processors or may be integrated with the processor 552.

[66] The processor 552 is preferably connected to a communication bus 554. The communication bus 554 may include a data channel for facilitating information transfer between storage and other peripheral components of the computer system 550. The communication bus 554 further may provide a set of signals used for communication with the processor 552, including a data bus, address bus, and control bus (not shown). The communication bus 554 may comprise any standard or non-standard bus architecture such as, for example, bus architectures compliant with industry standard architecture ("ISA"), extended industry standard architecture ("EISA"), Micro Channel Architecture ("MCA"), peripheral component interconnect ("PCI") local bus, or standards promulgated by the Institute of Electrical and Electronics Engineers ("IEEE") including IEEE 488 general-purpose interface bus ("GPIB"), IEEE 696/S-100, and the like.

[67] Computer system 550 preferably includes a main memory 556 and may also include a secondary memory 558. The main memory 556 provides storage of instructions and data for programs executing on the processor 552. The main memory 556 is typically semiconductor-based memory such as dynamic random access memory ("DRAM") and/or static random access memory ("SRAM"). Other semiconductor-based memory types include, for example, synchronous dynamic random access memory ("SDRAM"), Rambus dynamic random access memory ("RDRAM"), ferroelectric random access memory ("FRAM"), and the like, including read only memory ("ROM").

[68] The secondary memory 558 may optionally include a hard disk drive 560 and/or a removable storage drive 562, for example a floppy disk drive, a magnetic tape drive, a compact disc ("CD") drive, a digital versatile disc ("DVD") drive, etc. The removable storage drive 562 reads from and/or writes to a removable storage medium 564 in a well-known manner. Removable storage medium 564 may be, for example, a floppy disk, magnetic tape, CD, DVD, etc.

[69] The removable storage medium 564 is preferably a computer readable medium having stored thereon computer executable code (i.e., software) and/or data. The computer software or data stored on the removable storage medium 564 is read into the computer system 550 as electrical communication signals 578.

[70] In alternative embodiments, secondary memory 558 may include other similar means for allowing computer programs or other data or instructions to be loaded into the computer system 550. Such means may include, for example, an external storage medium 572 and an interface 570. Examples of external storage medium 572 may include an external hard disk drive or an external optical drive, or and external magneto-optical drive.

[71] Other examples of secondary memory 558 may include semiconductor-based memory such as programmable read-only memory ("PROM"), erasable programmable read-only memory ("EPROM"), electrically erasable read-only memory ("EEPROM"), or flash memory (block oriented memory similar to EEPROM). Also included are any other removable storage units 572 and interfaces 570, which allow software and data to be transferred from the removable storage unit 572 to the computer system 550.

[72] Computer system 550 may also include a communication interface 574. The communication interface 574 allows software and data to be transferred between computer system 550 and external devices (e.g. printers), networks, or information sources. For example, computer software or executable code may be transferred to computer system 550 from a network server via communication interface 574. Examples of communication interface 574 include a modem, a network interface card ("NIC"), a

communications port, a PCMCIA slot and card, an infrared interface, and an IEEE 1394 fire-wire, just to name a few.

[73] Communication interface 574 preferably implements industry promulgated protocol standards, such as Ethernet IEEE 802 standards, Fiber Channel, digital subscriber line ("DSL"), asynchronous digital subscriber line ("ADSL"), frame relay, asynchronous transfer mode ("ATM"), integrated digital services network ("ISDN"), personal communications services ("PCS"), transmission control protocol/Internet protocol ("TCP/IP"), serial line Internet protocol/point to point protocol ("SLIP/PPP"), and so on, but may also implement customized or non-standard interface protocols as well.

[74] Software and data transferred via communication interface 574 are generally in the form of electrical communication signals 578. These signals 578 are preferably provided to communication interface 574 via a communication channel 576. Communication channel 576 carries signals 578 and can be implemented using a variety of communication means including wire or cable, fiber optics, conventional phone line, cellular phone link, radio frequency (RF) link, or infrared link, just to name a few.

[75] Computer executable code (i.e., computer programs or software) is stored in the main memory 556 and/or the secondary memory 558. Computer programs can also be received via communication interface 574 and stored in the main memory 556 and/or the secondary memory 558. Such computer programs, when executed, enable the computer system 550 to perform the various functions of the present invention as previously described.

[76] In this description, the term "computer readable medium" is used to refer to any media used to provide computer executable code (e.g., software and computer programs) to the computer system 550. Examples of these media include main memory 556, secondary memory 558 (including hard disk drive 560, removable storage medium 564, and external storage medium 572), and any peripheral device communicatively coupled with communication interface 574 (including a network information server or other

network device). These computer readable mediums are means for providing executable code, programming instructions, and software to the computer system 550.

[77] In an embodiment that is implemented using software, the software may be stored on a computer readable medium and loaded into computer system 550 by way of removable storage drive 562, interface 570, or communication interface 574. In such an embodiment, the software is loaded into the computer system 550 in the form of electrical communication signals 578. The software, when executed by the processor 552, preferably causes the processor 552 to perform the inventive features and functions previously described herein.

[78] Various embodiments may also be implemented primarily in hardware using, for example, components such as application specific integrated circuits ("ASICs"), or field programmable gate arrays ("FPGAs"). Implementation of a hardware state machine capable of performing the functions described herein will also be apparent to those skilled in the relevant art. Various embodiments may also be implemented using a combination of both hardware and software.

[79] While the particular systems and methods herein shown and described in detail are fully capable of attaining the above described objects of this invention, it is to be understood that the description and drawings presented herein represent a presently preferred embodiment of the invention and are therefore representative of the subject matter which is broadly contemplated by the present invention. It is further understood that the scope of the present invention fully encompasses other embodiments that may become obvious to those skilled in the art and that the scope of the present invention is accordingly limited by nothing other than the appended claims.